Name: Abdul Moeez
Registration ID: 241804

Database Systems

Semester Project

Submitted By:

Name: Abdul Moeez

Roll Number: 241804

Submitted To:

Name: Ms. Warda Aslam

Name: Abdul Moeez
Registration ID: 241804

Introduction

This project focuses on designing and implementing a comprehensive relational database for a music streaming service modeled after Spotify. The system manages a wide range of information, including users, artists, songs, playlists, albums, genres, languages, user behaviors (likes, recently played, search history), and system analytics. The objective is to support high user engagement and large-scale song libraries while ensuring data integrity, normalization, and query efficiency.

Scenario Overview

Users interact with the system by:

- Streaming songs
- Creating and managing playlists
- Following their favorite artists
- Liking and replaying songs
- Searching for specific songs, artists, or albums

Artists upload songs, publish albums, and monitor fan engagement. Behind the scenes, the database supports:

- Tracking listening statistics
- Logging search activity
- Managing device-specific playback

The system delivers the backend for personalized experiences such as recommendations and music discovery.

EERD (Enhanced Entity-Relationship Diagram)

The EERD outlines all key entities and relationships. Major entities include:

- User: Captures all basic user details. Users can be regular listeners or specialized as artists.
- Artist: A subtype of User with fields like bio and is verified. Artists can upload songs and manage albums.
- Song:
- Attributes: song_id, title, duration, release_date
- Linked to one artist, one album, one genre, and one language
- Album:
- Created by artists and can contain multiple songs
- Attributes: album_id, name, release_year
- Genre & Language:
- Used to classify songs
- Playlist:
- Created by users
- Has a many-to-many relationship with songs via playlist_songs

- Likes:
- Tracks which songs a user has liked
- Recently Played:
- Logs timestamped entries of songs recently streamed by users
- Listening Stats:
- Records play_count and total_duration per user-song pair
- Follow Artist:
- Maps users to the artists they follow
- Search History:
- Logs search input, search type, and timestamps
- User Devices:
- Captures device info like device name and operating system used to access the platform

All relationships are enforced via foreign keys. Many-to-many relationships are resolved using associative tables such as playlist_songs, likes, and follow_artist. The schema is fully normalized to 3NF.

Software and Tools Used

- Microsoft SQL Server Express: Backend RDBMS for implementing the schema
- SQL Server Management Studio (SSMS): Used for schema design, query testing, and debugging
- JavaFX (JFX): Frontend UI development for demonstration purposes
- IntelliJ IDEA: IDE used for integrating frontend with database
- JDBC: For connecting the JavaFX frontend to SQL Server database

Database Design & Implementation

- Database Creation: Tables were created with CREATE TABLE statements, defining primary and foreign keys.
- Data Insertion: Sample records were inserted using INSERT statements.
- Relationships: Enforced using foreign key constraints.
- Normalization: All tables were normalized to 3NF.

Stored Procedures & Triggers

- sp_insert_song: Inserts new songs and updates the corresponding album's metadata
- trg_update_stats: Trigger to update listening_stats table when a song is played
- sp_get_recommendations: Fetches recommended songs based on user play history and likes

Sample Queries

- SELECT TOP 10 title FROM Songs ORDER BY likes_count DESC – Most liked songs
- SELECT * FROM RecentlyPlayed WHERE user_id = X ORDER BY played_at DESC – User's recent streams
- SELECT SUM(total_duration) FROM ListeningStats WHERE user_id = X – Total listening time per user
- SELECT artist_name FROM FollowArtist WHERE user_id = X – Artists followed by a user

- SELECT * FROM Songs WHERE genre_id = X OR language_id = Y – Songs by genre/language

Frontend Integration with JavaFX

- A minimal JavaFX UI was created for demonstration
- Functionality includes:
- User login
- Song browsing
- Playlist creation
- Like/unlike toggle buttons
- Playback simulation (affects stats and recently played)

Conclusion

This project demonstrates the end-to-end process of designing, normalizing, and implementing a relational database system that supports a full-featured music streaming platform. With a JavaFX frontend and SQL Server backend, we successfully managed user interactions, artist content, and streaming data using SQL queries, stored procedures, and triggers. This experience deepened our understanding of real-world database application design and SQL programming.